Java Basic 01

Write Once Run Anywhere

Jupiter Zhuo

What’s Java

- Java is a programming language and a platform.
- Java is a high level, robust, secured and object-oriented programming language
- **Platform**: Any hardware or software environment in which a program runs, is known as a platform. Since Java has its own runtime environment (JRE) and API, it is called platform.

Where it is used?

1. Desktop Applications such as acrobat reader, media player, antivirus etc.
2. Web Applications
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games etc.

Types of Java Applications

Standalone Application

Web Application

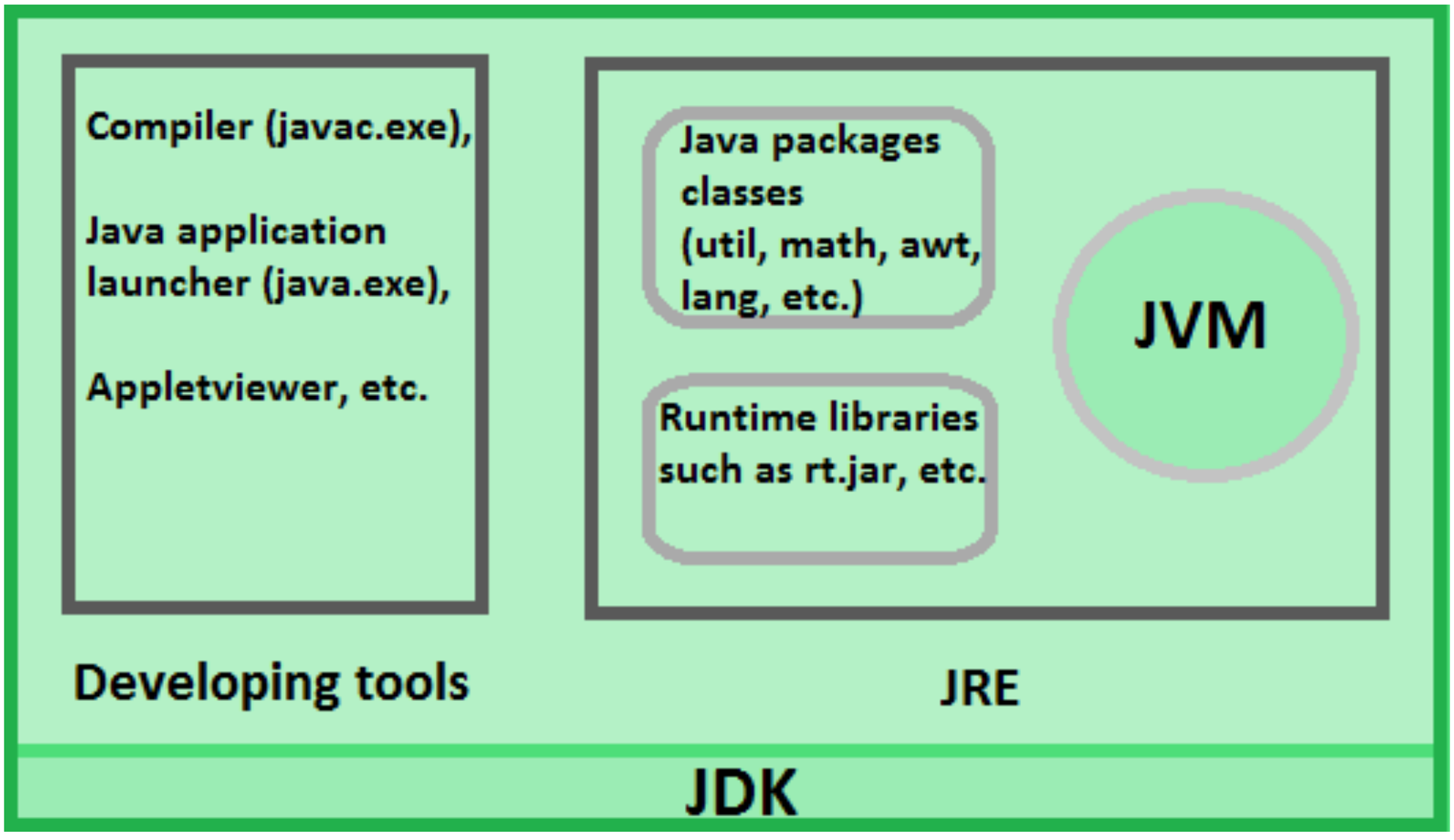
Enterprise Application

Mobile Application

Java Platforms / Editions

- **Java SE (Java Standard Edition)**
- **Java EE (Java Enterprise Edition)**
- **Java ME (Java Micro Edition)**
- **JavaFx**

Difference between JDK, JRE and JVM

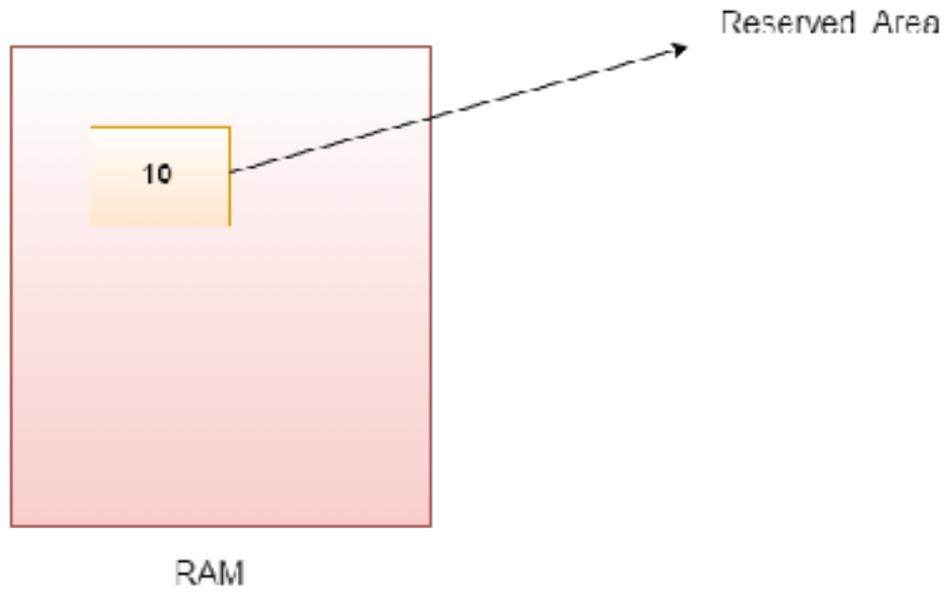


Java Basic Syntax

- Case Sensitivity
- Class Name Ex : MyFirstJavaClass
- Method Name Ex : myMethodName
- Program Name = Class Name
- public static void main(String args[]) = main method

Java Variable

Variable is name of *reserved area allocated in memory*. In other words, it is a *name of memory location*. It is a combination of "vary + able" that means its value can be changed.



Java Variable Naming Conventions

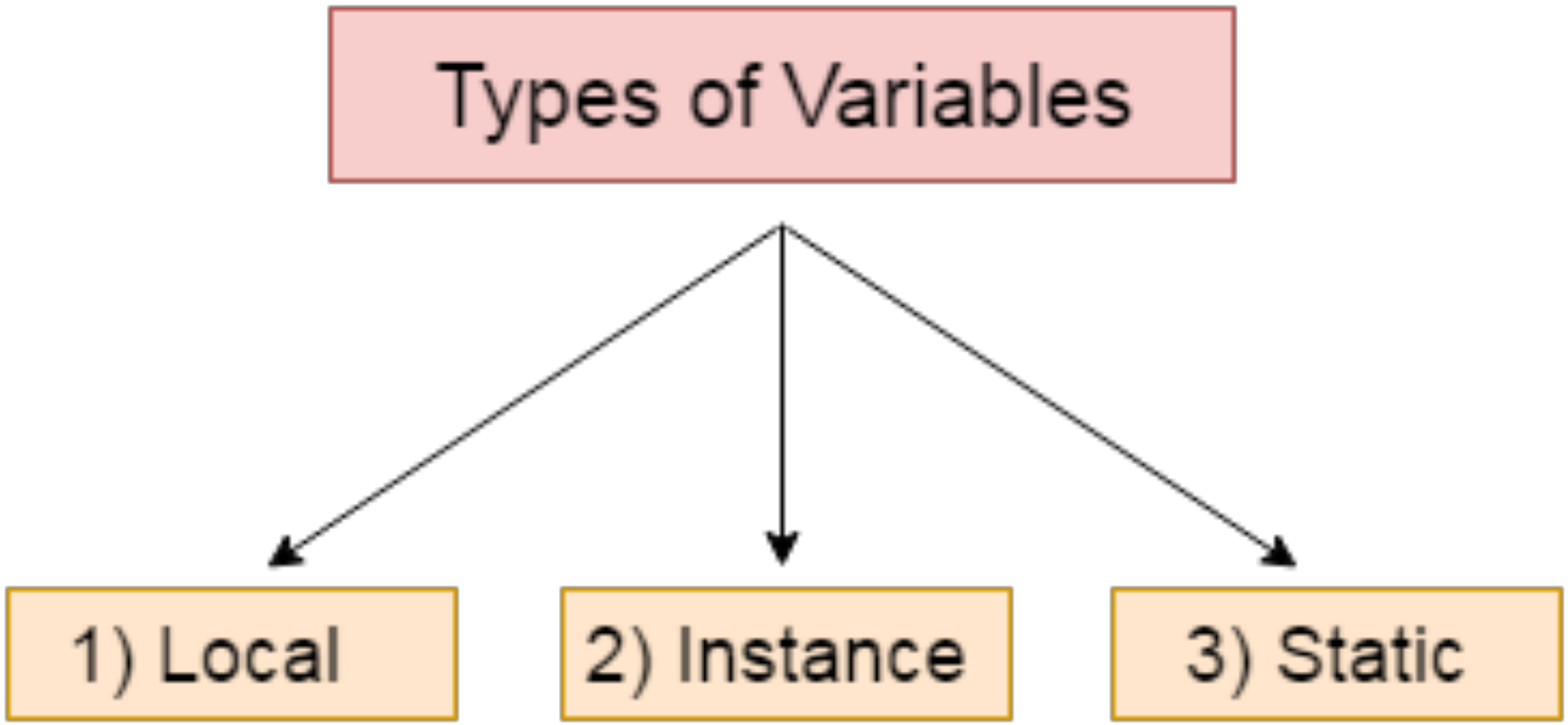
- **Java variable names are case sensitive. The variable name `money` is not the same as `Money` or `MONEY`.**

- **Java variable names must start with a letter, or the $ or _ character.**

- **After the first character in a Java variable name, the name can also contain numbers (in addition to letters, the $, and the _ character).**

- **Variable names cannot be equal to reserved key words in Java. For instance, the words `int` or `for` are reserved words in Java. Therefore you cannot name your variables `int` or `for`.**

Types of Variable



Java Indentifiers

All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_).

After the first character, identifiers can have any combination of characters.

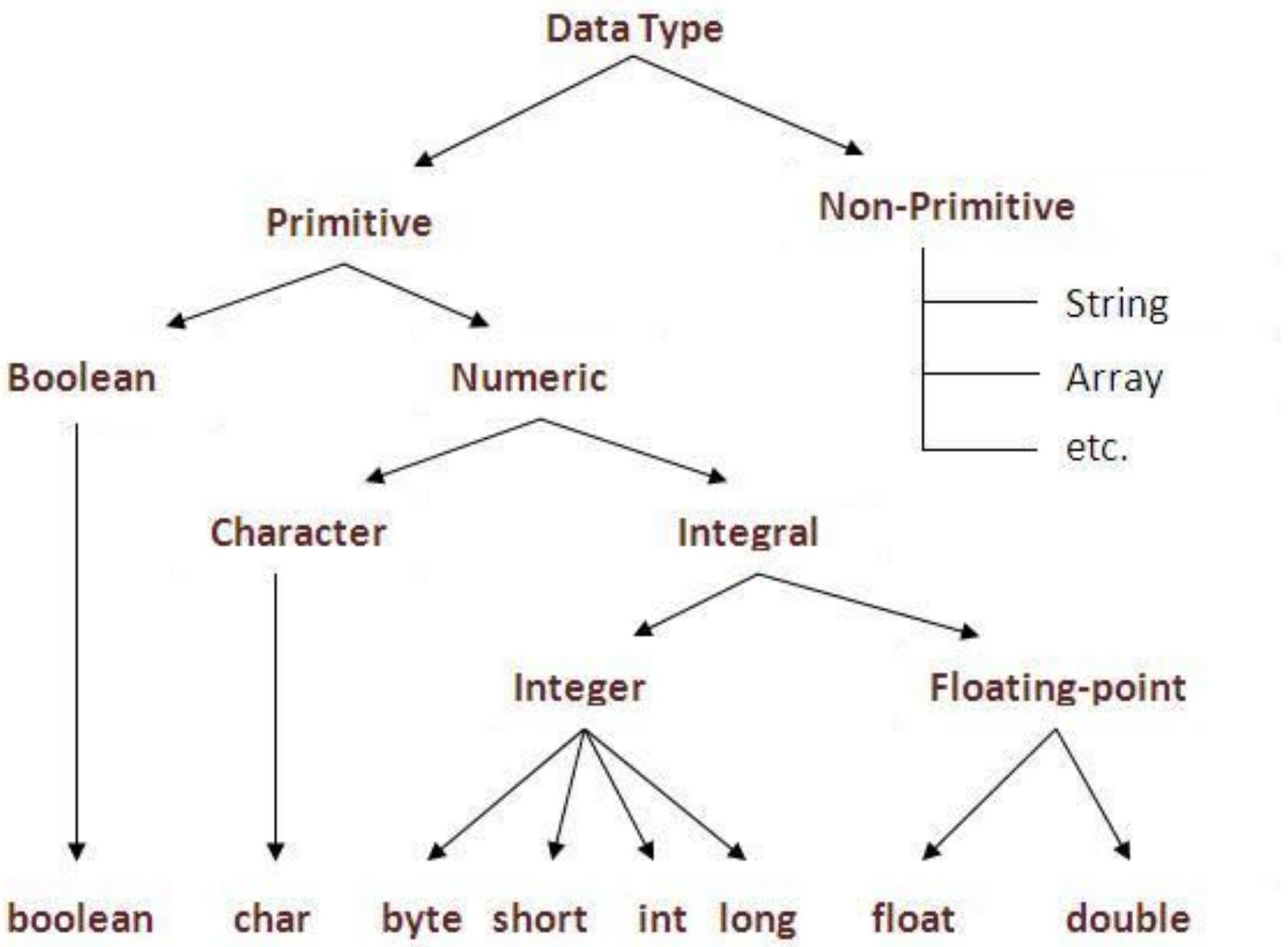
A key word cannot be used as an identifier.

Most importantly, identifiers are case sensitive.

Examples of legal identifiers: age, $salary, _value, __1_value.

Examples of illegal identifiers: 123abc, -salary.

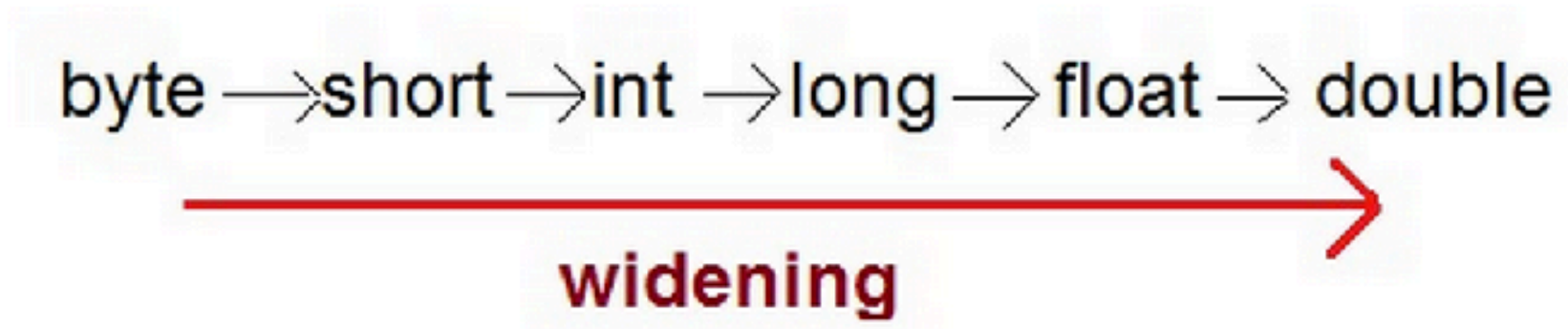
Java Data Type



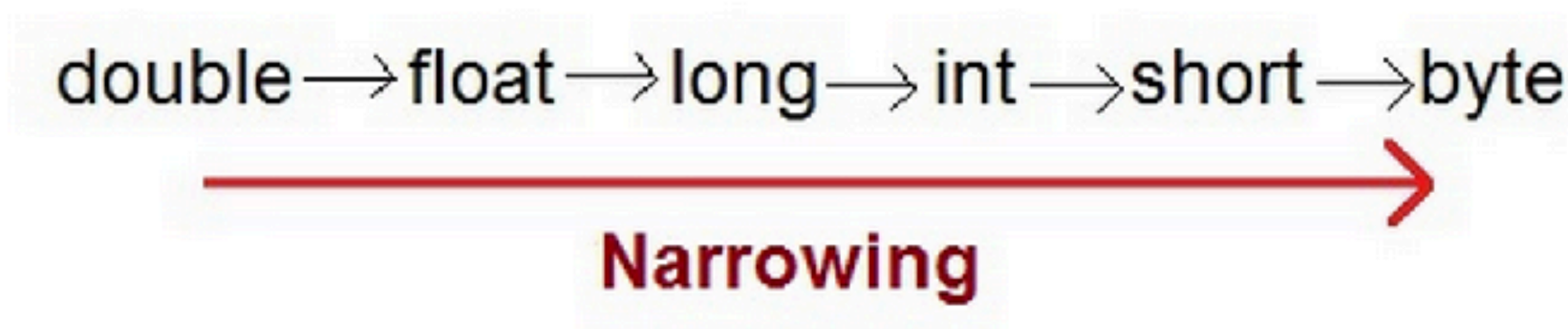
Java Type Casting

In Java, type casting is classified into two types,

- Widening Casting(Implicit)



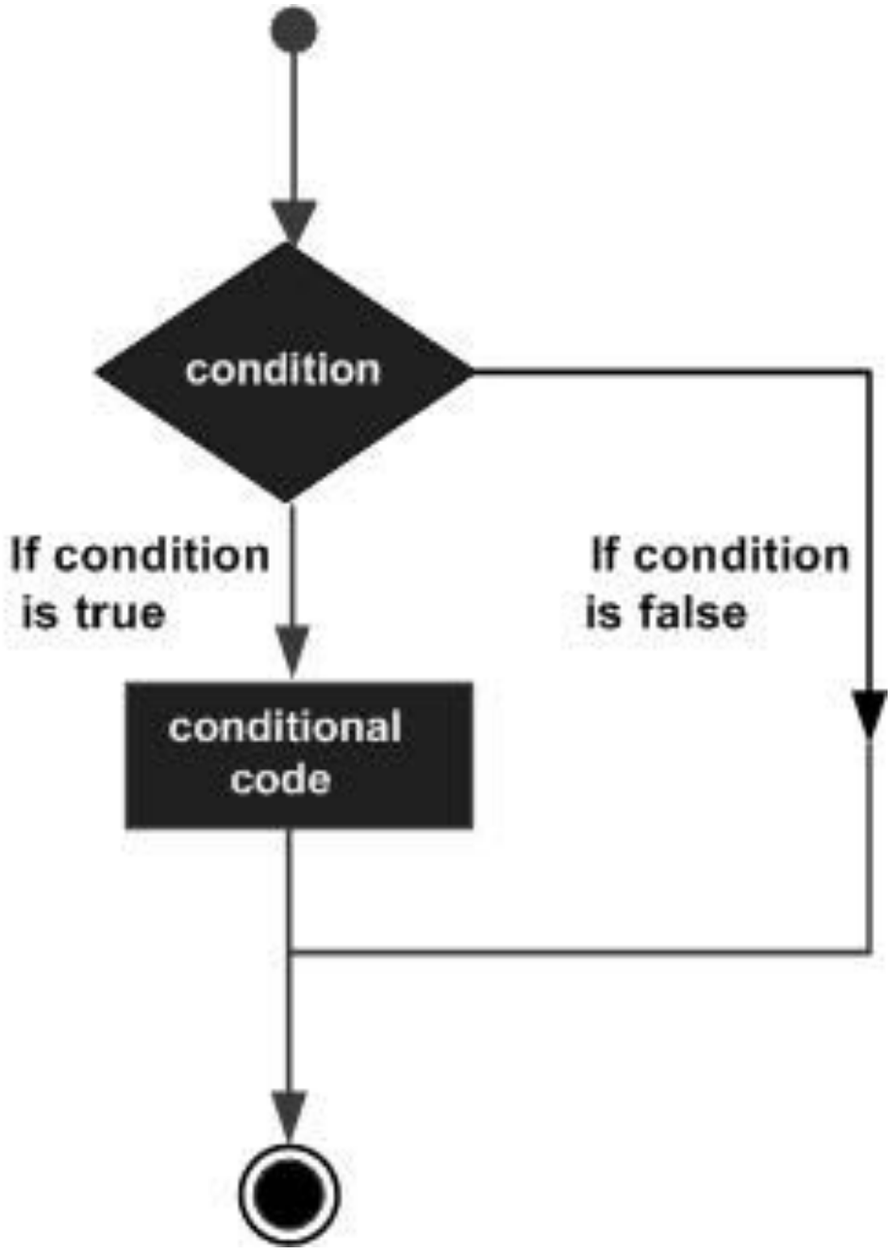
- Narrowing Casting(Explicitly done)



Java Operator

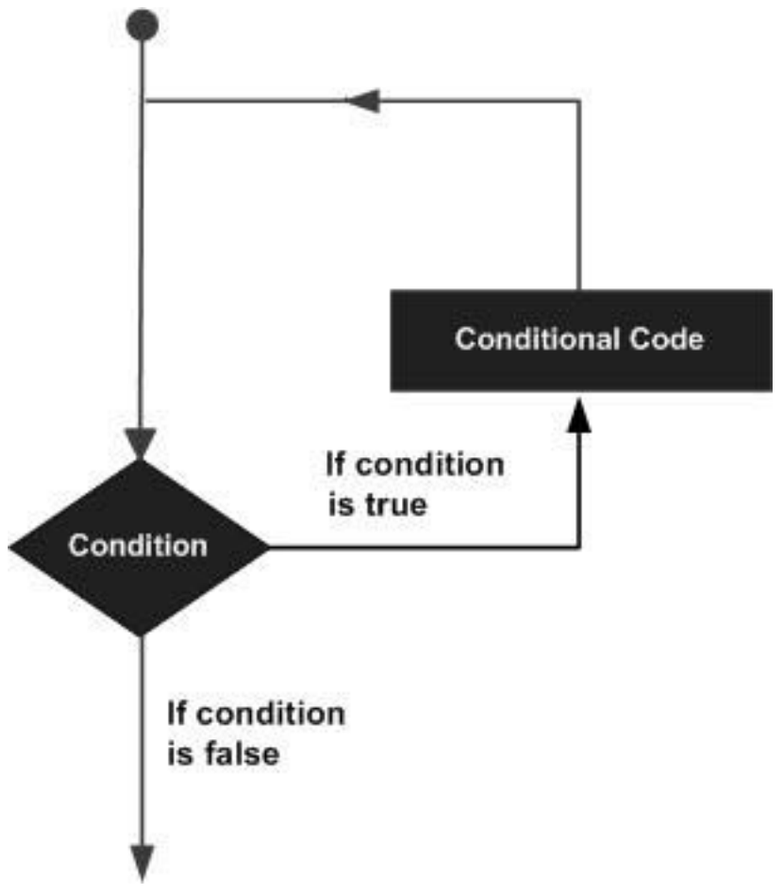
Operator Type	Category	Precedence
Unary	postfix	*expr*++ *expr*--
	prefix	++*expr* --*expr* +*expr* -*expr* ~ !
Arithmetic	multiplicative	* / %
	additive	+ -
Shift	shift	<< >> >>>
Relational	comparison	< > <= >= instanceof
	equality	== !=
Bitwise	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	|
Logical	logical AND	&&
	logical OR	||
Ternary	ternary	? :
Assignment	assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Control Statement



- If statement
- if… else statement
- nested if statement
- switch statement

Looping



- while loop
- do … while loop
- for loop

Latihan

- Soal 1

  Input data : 5

  1,3,5,7,9

  Jumlah dari Lima data yang diinput =25

- Soal 2

  Input data : 2

  2 x 0 = 0

  2 x 1 = 2

  2 x 2= 4

  Input data : 4

  4 x 0 = 0

  4 x 1 = 4

  4 x 2= 8

  4 x 3= 12

  4 x 4= 16

Array

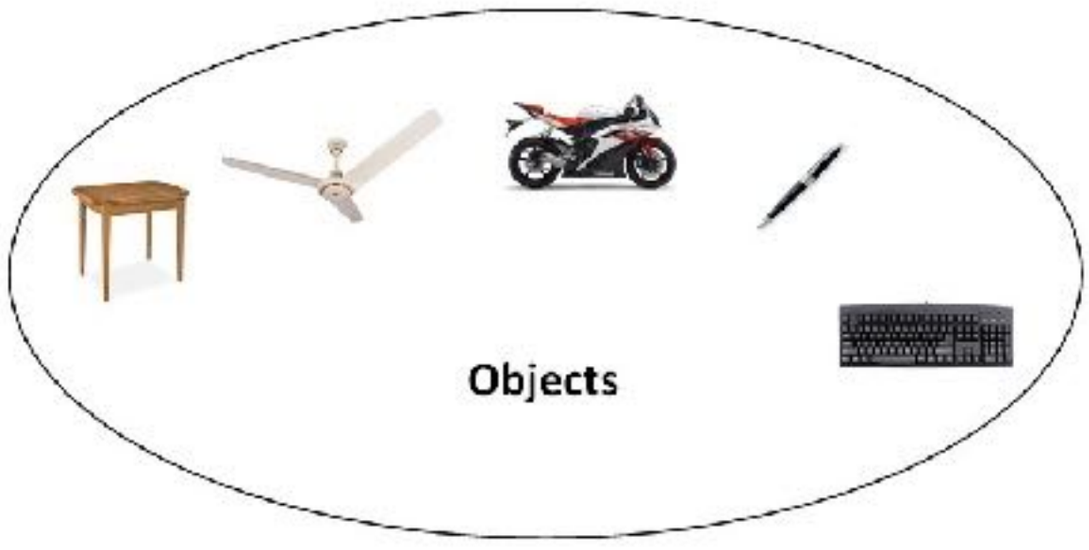
- Normally, array is a collection of similar type of elements that have contiguous memory location.
- **Java array** is an object the contains elements of similar data type. It is a data structure where we store similar elements. We can store only fixed set of elements in a java array.
  - Advantage of Java Array
    - Code Optimization
    - Random access
  - Disadvantage of Java Array
    - Size Limit

Java String

- In java, string is basically an object that represents sequence of char values. An array of characters works same as java string

Java OOP

- OOP - Object Oriented Programming
- OOP concepts
    - Object
    - Class
    - Inheritance
    - Polymorphism
    - Abstraction
    - Encapsulation



Java Class

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

A class in Java can contain:

- fields
- methods
- Constructor
- blocks
- nested class and interfaces

Access Modifiers

Modifier	class	constructor	method	Data/variables
public	Yes	Yes	Yes	Yes
protected		Yes	Yes	Yes
default	Yes	Yes	Yes	Yes
private		Yes	Yes	Yes
static			Yes	Yes
final	Yes		Yes	Yes

Access Modifier

Visibility	Public	Protected	*Default*	Private
From the same class	Yes	Yes	Yes	Yes
From any class in the same package	Yes	Yes	Yes	No
From a subclass in the same package	Yes	Yes	Yes	No
From a subclass outside the same package	Yes	Yes, *through inheritance*	No	No
From any non-subclass class outside the package	Yes	No	No	No

Constructor in Java

- **Constructor in java** is a *special type of method* that is used to initialize the object.
- invoked at the time of object creation
- Rules for creating java constructor
  - same name with class
  - constructor must have no explicit return type
- Type of Java Constructor
  - Default Constructor
  - Parameterized constructor

Constructor VS Method

Java Constructor	Java Method
Constructor is used to initialize the state of an object.	Method is used to expose behaviour of an object.
Constructor must not have return type.	Method must have return type.
Constructor is invoked implicitly.	Method is invoked explicitly.
The java compiler provides a default constructor if you don't have any constructor.	Method is not provided by compiler in any case.
Constructor name must be same as the class name.	Method name may or may not be same as class name.

Java Static keyword

The **static keyword** in java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than instance of the class.

Inheritance in Java

- **Inheritance in java** is a mechanism in which one object acquires all the properties and behaviors of parent object.

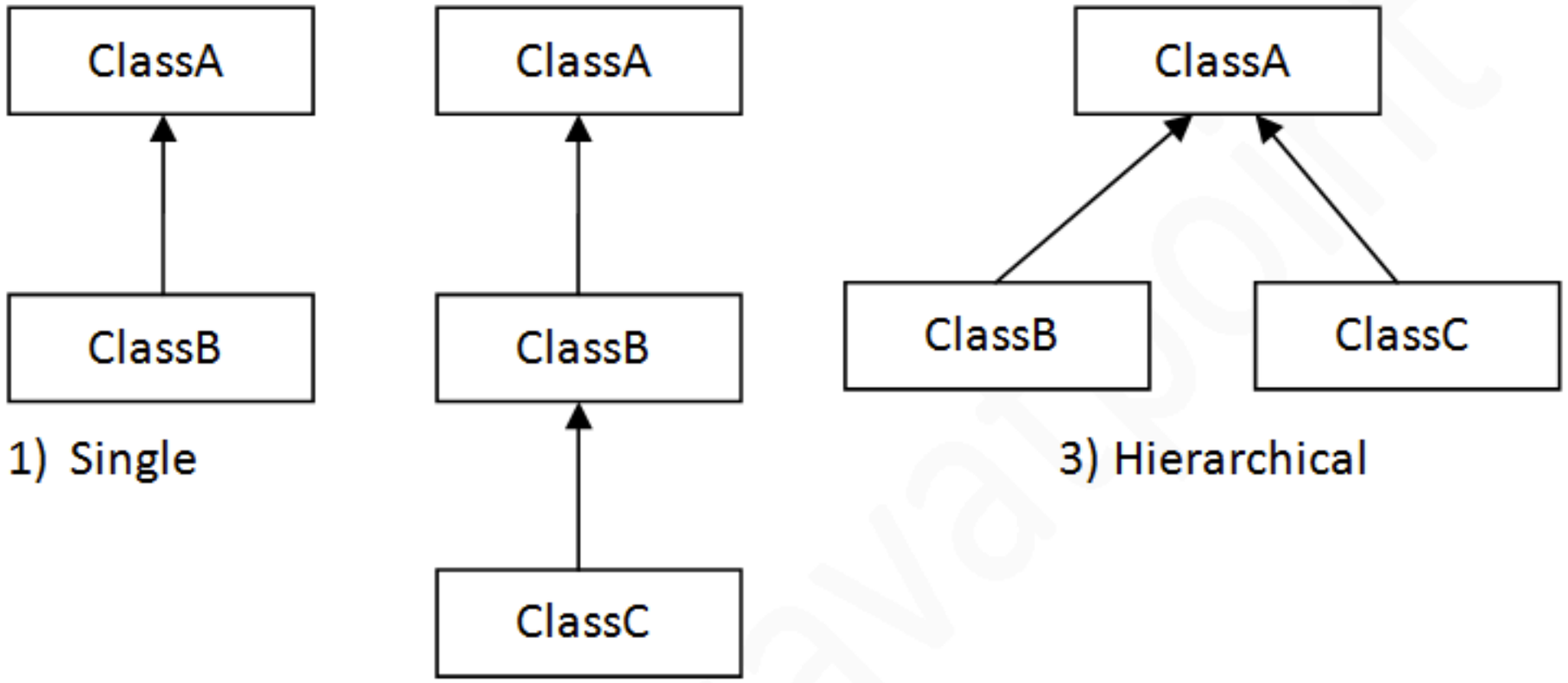
- Inheritance represents the **IS-A relationship**, also known as *parent-child* relationship.

- Why Use Inheritance In Java

  - Method overriding

  - Code Reusability

Type Of Java Inheritance



Method Overloading In java

- If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading**
- Advantage of method overloading
  - Method overloading increases the readability of the program.
  - Different ways to overload the method
- There are two ways to overload the method in java
  - By changing number of arguments
  - By changing the data type
- In java, Method Overloading is not possible by changing the return type of the method only.

Method Overriding in Java

- If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in java**.
- Usage of Java Method Overriding
  - Method overriding is used to provide specific implementation of a method that is already provided by its super class.
  - Method overriding is used for runtime polymorphism
- Rules for Java Method Overriding
  - method must have same name as in the parent class
  - method must have same parameter as in the parent class.
  - must be IS-A relationship (inheritance).

Overriding VS Overloading

No.	Method Overloading	Method Overriding
1)	Method overloading is used *to increase the readability* of the program.	Method overriding is used *to provide the specific implementation* of the method that is already provided by its super class.
2)	Method overloading is performed *within class*.	Method overriding occurs *in two classes* that have IS-A (inheritance) relationship.
3)	In case of method overloading, *parameter must be different*.	In case of method overriding, *parameter must be same*.
4)	Method overloading is the example of *compile time polymorphism*.	Method overriding is the example of *run time polymorphism*.
5)	In java, method overloading can't be performed by changing return type of the method only. *Return type can be same or different* in method overloading. But you must have to change the parameter.	*Return type must be same or covariant* in method overriding.

super keyword in java

- The **super** keyword in java is a reference variable which is used to refer immediate parent class object.

- Usage of Java super Keyword

  - super can be used to refer immediate parent class instance variable.

  - super can be used to invoke immediate parent class method.

  - super() can be used to invoke immediate parent class constructor.

Final Keyword In Java

The **final keyword** in java is used to restrict the user. The java final keyword can be used in many context. Final can be:

1. variable (Constanta)
2. method (Can’t override it)
3. class (Can’t extend it)

Polymorphism in Java

Polymorphism in java is a concept by which we can perform a *single action by different ways*. Polymorphism is derived from 2 greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.

There are two types of polymorphism in java: **compile time polymorphism** and **runtime polymorphism.** We can perform polymorphism in java by method overloading and method overriding.

Runtime Polymorphism in Java

Runtime polymorphism or **Dynamic Method Dispatch** is a process in which a call to an overridden method is resolved at runtime rather than compile-time.

In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

Encapsulation in Java

Encapsulation in java is a process of wrapping code and data together into a single unit

We can create a fully encapsulated class in java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it.

Advantage of Encapsulation in java

1. By providing only setter or getter method, you can make the class **read-only or write-only**.

2. It provides you the **control over the data**. Suppose you want to set the value of id i.e. greater than 100 only, you can write the logic inside the setter method.

Abstract Class in Java

A class that is declared with abstract keyword, is known as abstract class in java. It can have abstract and non-abstract methods (method with body).

Abstraction is a process of hiding the implementation details and showing only functionality to the user. Another way, it shows only important things to the user and hides the internal details for example sending sms, you just type the text and send the message. You don't know the internal processing about the message delivery.Abstraction lets you focus on what the object does instead of how it does it.

Interface In Java

An **interface in java** is a blueprint of a class. It has static constants and abstract methods.

The interface in java is **a mechanism to achieve abstraction**. There can be only abstract methods in the java interface not method body. It is used to achieve abstraction and multiple inheritance in Java.

Java Interface also represents IS-A relationship.

It cannot be instantiated just like abstract class.

Abstract VS Interface

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritance**.	Interface **supports multiple inheritance**.
3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
4) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
5) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.

6) **Example:**
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

Java Date

Letter	Description	Examples
y	Year	2013
M	Month in year	July, 07, 7
d	Day in month	1-31
E	Day name in week	Friday, Sunday
a	Am/pm marker	AM, PM
H	Hour in day	0-23
h	Hour in am/pm	1-12
m	Minute in hour	0-60
s	Second in minute	0-60